

1 1. A method comprising:
2 at a first point in a program in a computer programming language having dynamic
3 types and overloaded functions, constructing , using a function name, a function data
4 structure; the function data structure comprising information leading to a set of functions
5 visible at the first point;
6 at a second point, applying the function data structure to an argument list, the
7 applying comprising selecting a function using the function data structure and calling the
8 selected function.

1 2. The method of claim 1 wherein the constructing occurs within a first
2 scope, and wherein the applying the function data structure comprises selecting, from the
3 set of functions led to by the function data structure, the function that would be selected if
4 the function name were applied within the first scope to the argument list.

1 3. The method of claim 2 wherein the function data structure is applied in a
2 second scope that is different from the first scope.

1 4. The method of claim 1 wherein the information further comprises a
2 pointer which leads to native code.

1 5. The method of claim 4, wherein the information further comprises
2 information used by the native code.

1 6. The method of claim 1 wherein the information further comprises a
2 pointer which leads to native code which implements a top level function.

1 7. The method of claim 1 wherein the information further comprises a
2 pointer which leads to a mapping associating a type with native code which implements a
3 function.

1 8. The method of claim 1, wherein the information further comprises a
2 pointer leading to a lexical context, and further comprises a pointer leading to native code
3 which implements the function.

1 9. The method of claim 1, wherein the information further comprises a
2 pointer leading to interpreter code which implements the function.

1 10. The method of claim 1, wherein the information further comprises a
2 pointer which leads to a mapping associating a type with interpreter code for the function.

1 11. The method of claim 1, wherein the information further comprises a
2 pointer leading to a lexical context, and further comprises a pointer leading to interpreter
3 code which implements the function.

1 12. The method of claim 1, wherein the information further comprises
2 information which leads to the function name.

1 13. The method of claim 1 in which the information further comprises
2 information leading to an auxiliary function.

1 14. The method of claim 13 wherein the auxiliary function is selected from the
2 set consisting copy, delete, print and equal.

1 15. The method of claim 1, wherein the information further comprises
2 information used for storage management of the function data structure.

1 16. The method of claim 1 wherein the symbol "@" in front of the function
2 name in the computer programming language means to construct the function data
3 structure.

1 17. The method of claim 1 wherein the computer programming language
2 comprises a function named “feval” which, if applied to the function data structure and
3 the argument list, applies the function data structure to the argument list.

1 18. The method of claim 1, wherein the information further comprises a
2 pointer leading to a first auxiliary function;
3 applying the first auxiliary function to the function data structure and obtaining a
4 result from the applying.

1 19. The method of claim 18 wherein the first auxiliary function comprises at
2 least one selected from the set consisting of copy, delete, print and equal.

1 20. The method of claim 18 wherein the first auxiliary function comprises a
2 write function; and wherein applying the write function causes storing a first absolute
3 filename of a function lead to by the information in the function data structure.

1 21. The method of claim 20 wherein the information further comprises a
2 pointer leading to a read function and wherein applying the read function causes the first
3 absolute filename to be matched with a second absolute filename on the current path
4 which has the longest common tail with the first absolute filename.

1 22. The method of claim 18 wherein the information further comprises a pointer
2 leading to a second auxiliary function, and wherein applying the second auxiliary
3 function to the function data structure causes at least a portion of the information
4 contained in or pointed to by the function data structure to be returned as at least one
5 value.

1 23. The method of claim 1 wherein if the set of functions is other than a null set,
2 each member of the set of functions has the same name as the function name.

1 24. A computer program product, stored in a computer readable medium,
 2 comprising instructions to cause a computer to:
 3 at a first point, construct a function data structure using a function name; the
 4 function data structure comprising information leading to a set of functions visible at the
 5 first point;
 6 at a second point, apply the function data structure to an argument list selecting a
 7 function from the function data structure and call the selected function.

1 25. The computer program product of claim 24 wherein the construction
 2 occurs within a first scope, and wherein the application of the function data structure
 3 comprises selecting, from the set of functions led to by the function data structure, the
 4 function that would be selected if the function name were applied within the first scope to
 5 the argument list.

1 26. The computer program product of claim 25 wherein the function data
 2 structure is applied in a second scope that is different from the first scope.

1 27. The computer program product of claim 24 wherein the information
 2 further comprises a pointer which leads to native code.

1 28. The computer program product of claim 27, wherein the information
 2 further comprises information used by the native code.

1 29. The computer program product of claim 24 wherein the information
 2 further comprises a pointer which leads to native code which implements a top level
 3 function.

1 30. The computer program product of claim 24 wherein the information
 2 further comprises a pointer which leads to a mapping associating a type with native code
 3 which implements a function.

1 31. The computer program product of claim 24, wherein the information
2 further comprises a pointer leading to a lexical context, and further comprises a pointer
3 leading to native code which implements the function using the lexical context.

1 32. The computer program product of claim 24, wherein the information
2 further comprises a pointer leading to interpreter code which implements the function.

1 33. The computer program product of claim 24, wherein the information
2 further comprises a pointer which leads to a mapping associating a type with interpreter
3 code for the function.

1 34. The computer program product of claim 24, wherein the information
2 further comprises a pointer leading to a lexical context, and further comprises a pointer
3 leading to interpreter code which implements the function.

1 35. The computer program product of claim 24, wherein the information
2 further comprises information which leads to the function name.

1 36. The computer program product of claim 24 in which the information
2 further comprises information leading to an auxiliary function.

1 37. The computer program product of claim 36 wherein the auxiliary function
2 is selected from the set consisting copy, delete, print and equal.

1 38. The computer program product of claim 24, wherein the information
2 further comprises information used for storage management of the function data
3 structure.

1 39. The computer program product of claim 24 wherein the symbol "@" in
2 front of the function name in the instructions causes the computer to construct the
3 function data structure.

1 40. The computer program product of claim 24 wherein the computer
2 programming language comprises a function named “feval” which, if applied to the
3 function data structure and the argument list, causes the function data structure to be
4 applied to the argument list.

1 41. The computer program product of claim 24 wherein the information
2 further comprises a pointer leading to a first auxiliary function;
3 and wherein the instructions cause the first auxiliary function to be applied to the
4 function data structure and obtain a result from the applying.

1 42. The computer program product of claim 41 wherein the first auxiliary
2 function comprises at least one selected from the set consisting of copy, delete, print and
3 equal.

1 43. The computer program product of claim 41 wherein the first auxiliary
2 function comprises a write function; and wherein application of the write function causes
3 storing data on an external medium, the stored data comprising a first absolute filename
4 of a function lead to by the information in the function data structure.

1 44. The computer program product of claim 43 wherein the information
2 further comprises information leading to a read function, and wherein applying the read
3 function to the stored data causes the first absolute filename to be matched with a second
4 absolute filename on the current path which has the longest common tail with the first
5 absolute filename.

1 45. The computer program product of claim 41 wherein the information
2 further comprises a pointer leading to a second auxiliary function, and wherein the
3 instructions further cause a computer to, when applying the second auxiliary function to
4 the function data structure, cause at least a portion of the information contained in or
5 pointed to by the function data structure to be returned as at least one value.

- 1 46. The computer program product of claim 24 wherein if the set of functions
- 2 is other than a null set, each member of the set has the same name as the function name.